We claim:

1. A method of processing an application request on an end user application and an application server comprising the steps of:

a) initiating the application request on the end user application in a first language with a first application program;

b) transmitting the application request to the server and converting the application request from the first language of the first end user application to COBOL running on the application server;

c) processing said application request on the application server;

d) transmitting a response to the application request from the application server to the end user application, and converting theresponse to the application request from COBOL running on the application server to the first language of the first end user application; and

e) wherein the end user application and the application server have at least one connector therebetween, and the steps of (i) converting the application request from the first language of the first end user application as a source language to the COBOL running on the application server as a target language, and (ii) converting a response to the application request from the COBOL running on the application server as a source language to the first language of the first end user application as a target language, each comprise the steps of:

1) invoking connector metamodels of respective source and COBOL target languages;

2) populating the connector metamodels with metamodel data of each of the respective source and COBOL target languages; and

3) converting the source language to the COBOL arget language.

2. The method of claim 1 wherein the end user application is a web browser.

3. The method of claim 2 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

4. The method of claim 1 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

5. The method of claim 4 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

6. The method of claim 4 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

7. The method of claim 4 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

8. The method of claim 7 wherein the language metamodel metadata includes mappings between source and target languages.

9. The method of claim 8 wherein the source language is object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

10. The method of claim 9 wherein the language metamodel metadata maps object inheritances into references and pointers

11. The method of claim 4 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

12. The method of claim 1 wherein the transaction is a rich transaction comprising a plurality of individual transactions, and further comprising processing the plurality of individual transactions on one end user application and a plurality of application servers.

13. The method of claim 12 comprising passing individual transactions among individual application servers.

14. A transaction processing system comprising a client, a server, and at least one connector therebetween,

a) the client having an end user application, and being controlled and configured to initiate an application request with the server in a first language with a first application program and to transmit the application request to the server;

b) the connector being configured and controlled to receive the application request from the client, convert the application request from the first language of the first end user application running on the client to COBOL language running on the server;

c) the server being configured and controlled to receive the converted application request from the connector and processing the said application request in COBOL language with a second application program residing on the server, and to thereafter transmit a response to the application request through the connector back to the first application program on the client;

d) the connector being configured and controlled to receive a response to the application request from the server, to convert a response to the application request from the COBOL language running on the application server to the first language of the first application program running on the client; and

e) wherein connector between the client and the server is configured and controlled to (i) convert the application request from the first language of the client application on the client as a source language to the COBOL language running on the application server as a target language, and (ii) convert the response to the application request from the COBOL language running on the application server as a source language to the first language of the client application running on the client as a target language, each by a method comprising the steps of:

1) retrieving connector metamodels of respective source and target languages from a metamodel metadata repository;

2) populating the connector metamodels with metamodel data from the metamodel metadata repository for each of the respective source and target languages; and

3) invoking the retrieved, populated connector metamodels and converting the source language to the target language.

15. The system of claim 14 wherein the end user application is a web browser.

16. The system of claim 15 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

17. The system of claim 14 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

18. The system of claim 17 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

19. The system of claim 17 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

20. The system of claim 17 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

21. The system of claim 20 wherein the language metamodel metadata includes mappings between source and target languages.

22. The system of claim 23 wherein the source language is object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

23. The system of claim 22 wherein the language metamodel metadata maps object inheritances into references and pointers

24. The system of claim 18 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

25. The system of claim 14 wherein said system has a plurality of application servers and is configured and controlled to process rich transactions.

26. A transaction processing system configured and controlled to interact with a client application, and comprising a COBOL server, and at least one connector between the server and the client application, where the client has an end user application, and is controlled and configured to initiate an application request with the server in a first language with a first application program and to transmit theapplication request to the server, wherein:

a) the connector being configured and controlled to receive an application request from the client, convert the application request from the first language of the first end user application running on the client to the COBOL language running on the server;

b) the server being configured and controlled to receive the converted application request from the connector and process the said application request in the COBOL language with a second application program residing on the server, and to thereafter transmit a response to the application request through the connector back to the first application program on the client;

c) the connector being configured and controlled to receive the application request from the server, to convert a response to the application request from the COBOLlanguage running on the application server to the first language of the first application program running on the client; and

d) wherein connector between the client and the server is configured and controlled to (i) convert the application request from the first language of the client application on the client as a source language to the COBOL anguage running on the application server as a target language, and (ii) convert the response to the application request from the COBOL language running on the application server as a source language to the first language of the client application running on the client as a target language, each by a method comprising the steps of:

1) retrieving connector metamodel metadata of respective source and target languages from a metamodel metadata repository;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages from the metamodel metadata repository and invoking the retrieved, populated connector metamodels; and

3) converting the source language to the target language.

27. The system of claim 26 wherein the end user application is a web browser.

28. The system of claim 27 wherein the end user application is connected to the application server through a web server, and the web server comprises an connector.

29. The system of claim 26 wherein the metamodel metadata comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

30. The system of claim 29 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

31. The system of claim 29 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

32. The system of claim 29 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

33. The system of claim 32 wherein the language metamodel metadata includes mappings between source and target languages.

34. The system of claim 33 wherein the source language is object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

35. The method of claim 33 wherein the source language and the target language are different object oriented languages, and the language metamodel metadata maps encapsulated code and data between the languages.

36. The system of claim 33 wherein the language metamodel metadata maps object inheritances into references and pointers

37. The system of claim 29 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.

38. The system of claim 26 wherein said system has a plurality of application servers and is configured and controlled to process rich transactions.

39. A program product comprising a storage medium having invocation metamodel metadata, application domain interface metamodel metadata, and language metamodel metadata, and computer instructions for building a metamodel metadata repository of source and COBOL target language metamodel metadata.

40. The program product of claim 39 further comprising computer instructions for building connector stubs from said metamodel metadata.

41. The program product of claim 39 further comprising computer instructions to build a connector for carrying out the steps of:

1) retrieving connector metamodel metadata of respective source and target languages from the metamodel metadata repository;

2) populating the connector metamodels with metamodel data of each of the respective source and target languages from the metamodel metadata repository and invoking the retrieved, populated connector metamodels; and

3) converting the source language to the target language.

42. The program product of claim 41 wherein the metamodel metadata in the repository comprises invocation metamodel metadata, application domain interface metamodel metadata, and type descriptor metamodel metadata.

43. The program product of claim 42 wherein the invocation metamodel metadata is chosen from the group consisting of message control information, security data, transactional semantics, trace and debug information, pre-condition and post-condition resources, and user data.

44. The program product of claim 42 wherein the application domain interface metamodel metadata comprises input parameter signatures, output parameter signatures, and return types.

45. The program product of claim 42 wherein the application domain interface metamodel metadata further includes language metamodel metadata.

46. The program product of claim 45 wherein the language metamodel metadata includes mappings between source and target languages.

47. The program product of claim 46 wherein the source is object oriented, and the language metamodel metadata maps encapsulated objects into code and data.

48. The program product of claim 46 wherein the language metamodel metadata maps object inheritances into references and pointers

49. The program product of claim 43 wherein the type descriptor metamodel metadata defines physical realizations, storage mapping, data types, data structures, and realization constraints.